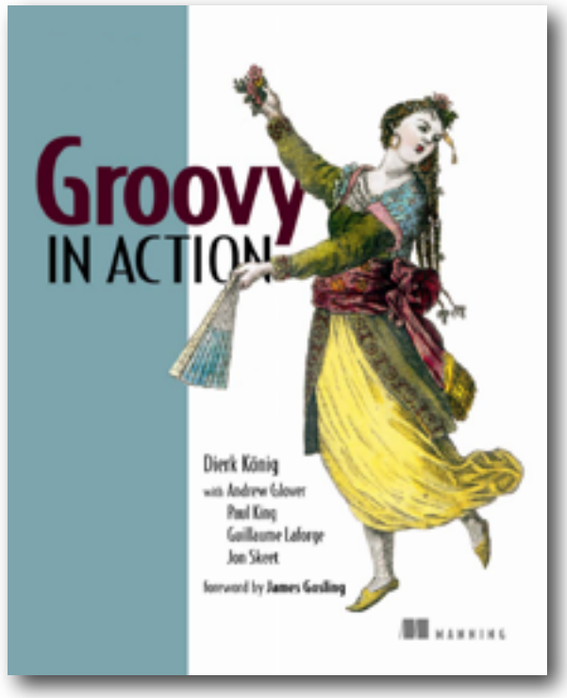
Google App Engine,
Groovy and Gaelyk

Gaelyk

Guillaume Laforge

- **Groovy Project Manager**
- JSR-241 Spec Lead
- Head of Groovy Development at **SpringSource**
- Initiator of the **Grails** framework



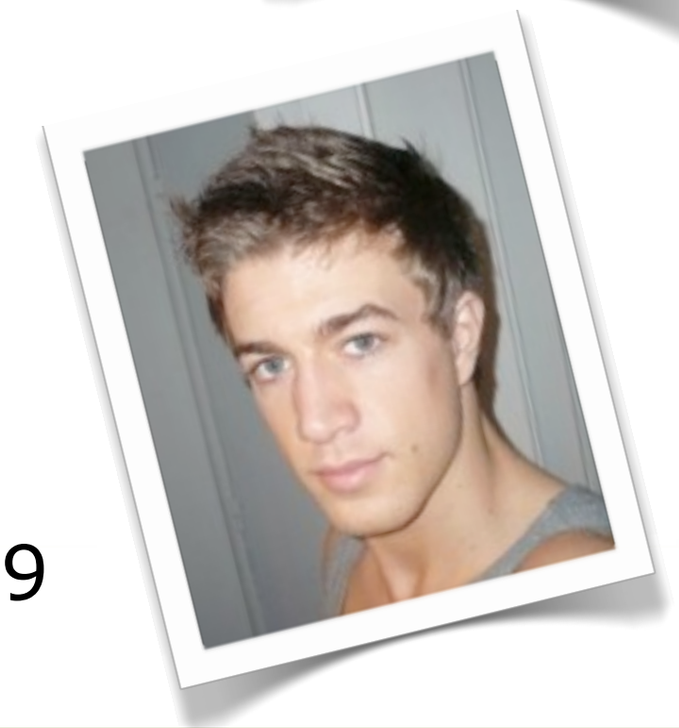
- Co-author of **Groovy in Action**

- Speaker: JavaOne, QCon, JavaZone, Sun TechDays, Devoxx, The Spring Experience, SpringOne, JAX, Dynamic Language World, IJTC, and more...

Marc-Antoine Guarrigue & Gaël Lazzari

- **Marc-Antoine**
  - Architecte et **Directeur de la R&D** chez **OCTO Technology** depuis 2001
    - Missions d'architecture d'applications Java EE
    - «Open Sourceur» dans l'âme
      - JCaptcha
      - Membre de l'OSSGTP
    - Developpement des sociétés
      - MassiveBrainGames.com
      - XDepend.com

- **Gaël Lazzari**
  - Consultant OCTO depuis septembre 2009



jeudi 12 novembre 2009

Cloud Computing

GÉNÉRIQUE ORIGINAL DE L'ÉMISSION TÉLÉVISÉE DE CHRISTOPHE IZARD

11.066

le village dans les nuages

tf1

disques Adès

45 T

GÉNÉRIQUE ORIGINAL DE L'ÉMISSION TÉLÉVISÉE DE CHRISTOPHE IZARD

11.066

~~le village~~ dans les nuages

tf1

disques Adès

45 T

GÉNÉRIQUE ORIGINAL DE L'ÉMISSION TÉLÉVISÉE DE CHRISTOPHE IZARD

11.066

le village dans les nuages

tf1

disques Adès 45 T

© TF1 1982

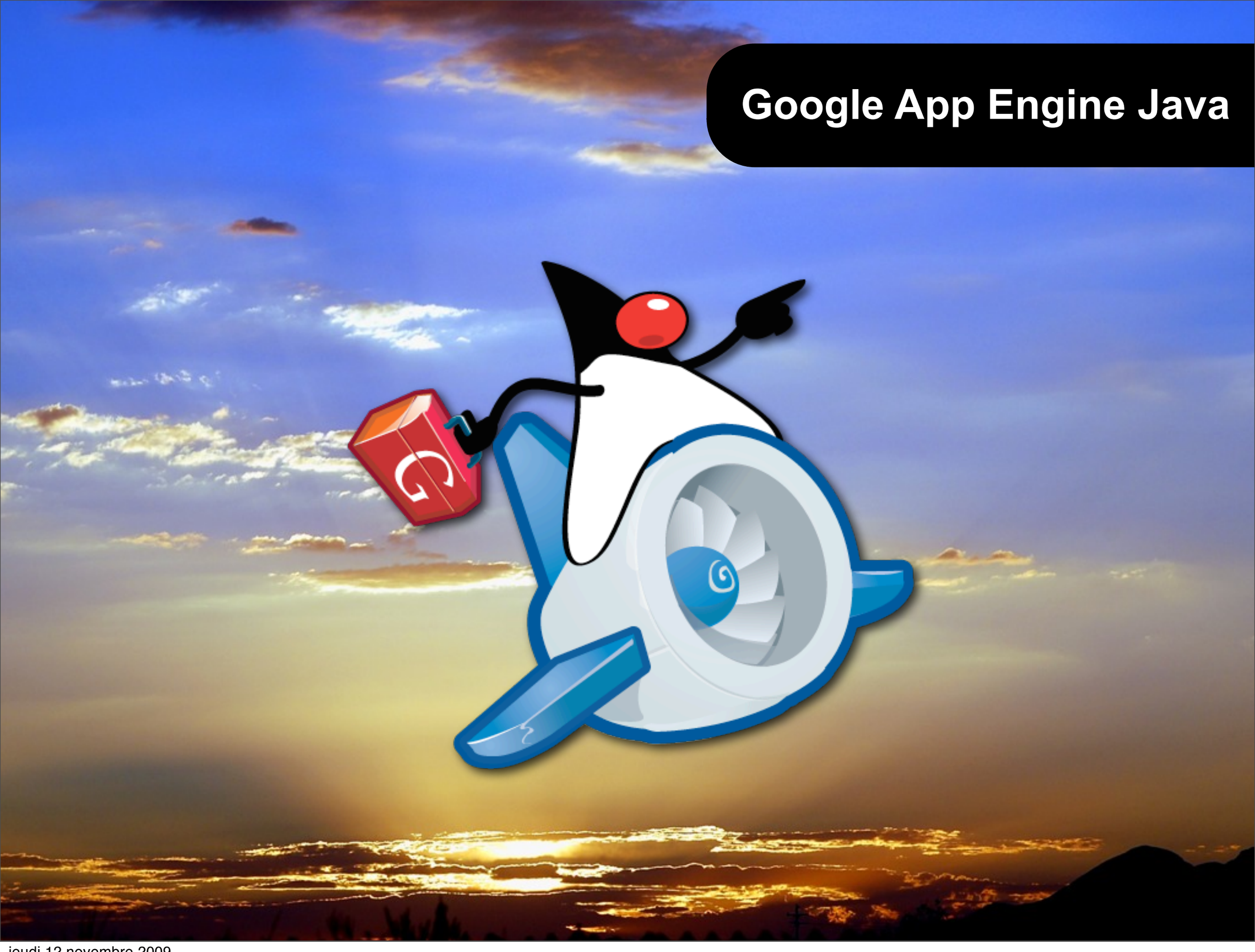
l'informatique

IaaS, PaaS, SaaS

- Software as a Service
  - Gmail, SalesForce.com
- Platform as a Service
  - Google App Engine
- Infrastructure as a Service
  - Amazon EC2


Google App Engine Java

Google App Engine Java



Google App Engine

- Google's PaaS solution
- Run your app on Google's infrastructure
- Initially just Python supported
- Since this year, **Java** supported too
  - Sandboxed **JVM**
  - Jetty **servlet container**
- Several JVM-compatible language supported
- A few services available
  - Email, XMPP, URL Fetch, Image, User, Task queues...

Key aspects

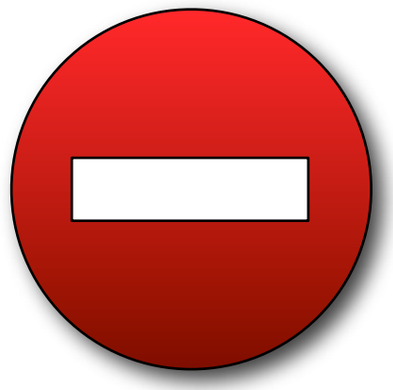
- You can use most of your usual web frameworks for developping apps on App Engine Java
  - **A WAR file, basically!**
- No OS image, or software to install
  - Unlike with Amazon EC2
- All the scaling aspects are handled for you
  - Database / session replication, load balancing
- There are quotas, but you need a high traffic application to start being charged
  - **Free to get started**

Available services

- **Memcache**
  - JCache implementation
  - Save on CPU and DB
- **URL Fetch**
  - Access remote resources
  - HttpUrlConnection
- **Mail**
  - Support both incoming and outgoing emails
- **Images**
  - Resize, crop, rotate...
- **XMPP**
  - Send / receive Jabber messages (GTalk)
- **User**
  - Use Google's user/ authentication system
- **Cron & Task queues**
  - Schedule tasks at regular intervals
  - Queue units of work

Limitations

- **Not our usual relational database**
  - key / value datastore
- **30 seconds request duration limit**
- Forbidden to
  - write on the file system
  - create threads
  - use raw sockets
  - issue system calls
  - use IO / Swing / etc. directly
    - There's a whitelist of classes allowed



- Number of files and their size are limited

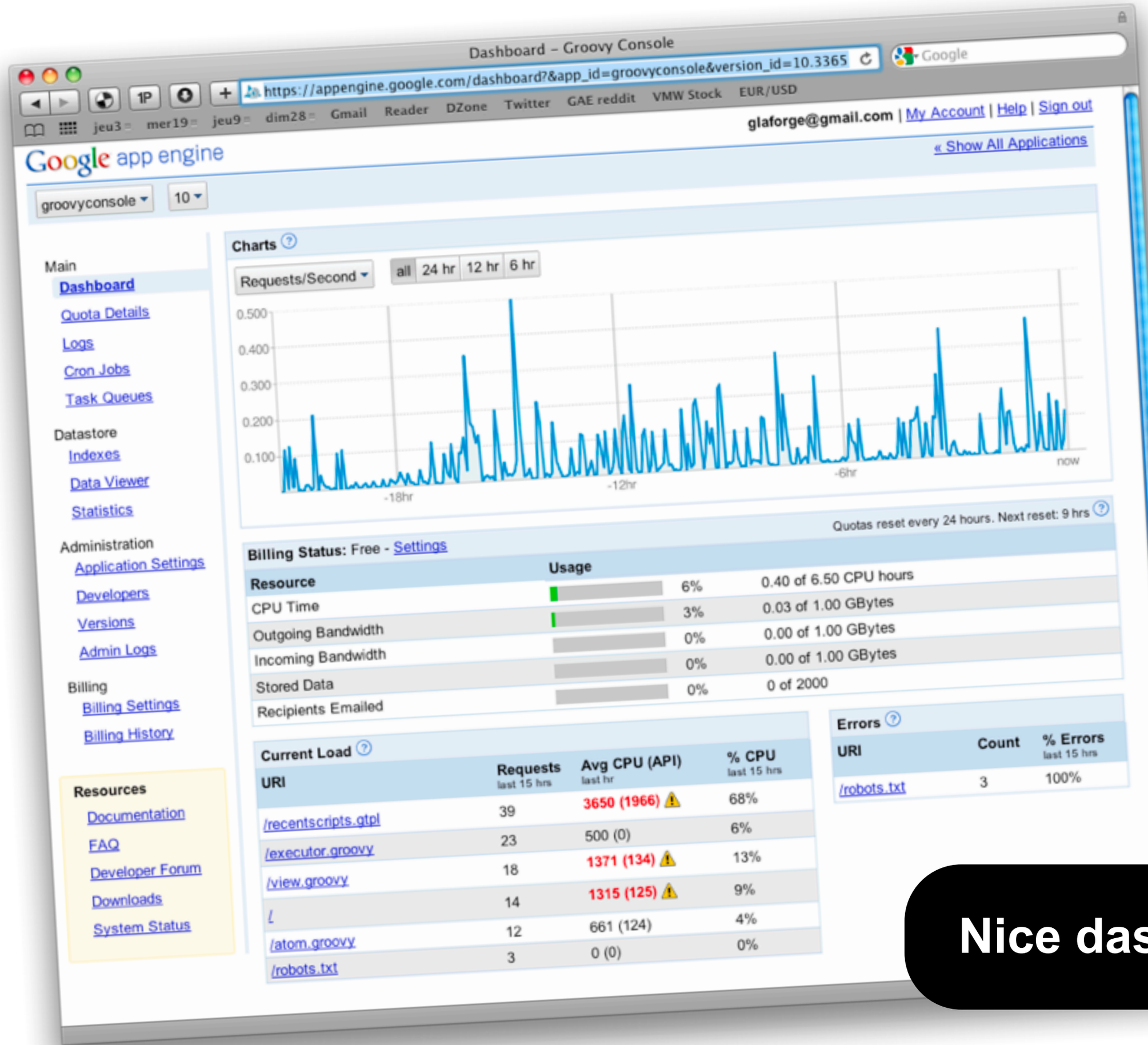
Quotas

Quotas (1/2)

- **Bandwidth**
  - 1,3M requests/day
  - 1GB/day in/out
  - 6.5 CPU hours/day
- **Datastore**
  - 10M calls
  - 1GB/day
  - 12GB in / 115GB out
  - 60 CPU hours/day
- **Mail**
  - 7K calls/day
  - 2K recepients/day
  - 5K emails/day
  - 2K attachments
  - 100MB of attachments
- **URL Fetch**
  - 657K calls/day
  - 4GB in/out /day

Quotas (2/2)

- **XMPP**
  - 657K calls/day
  - 4GB data sent/day
  - 657K recepients/day
  - 1MB invitations/day
- **Image manipulation**
  - 864 calls/day
  - 1GB in / 5GB out
  - 2.5M transforms
- **Memcache**
  - 8.6M calls/day
  - 10GB in
  - 50GB out
- **Task queues**
  - 100K calls



Nice dashboard

The Datastore

The datastore...

- **It's not your father's relational database!**
  - **You're not even using SQL!**
- Distributed **key / value store**
  - Based on Google's «BigTable»
- Supporting
  - Transactions and partitioning
  - Hierarchies through entity groups
- **Schema-less** approach
- Data access APIs
  - JPA and JDO
  - Direct low-level APIs

...and its «limitations»

- You're not using SQL
  - **No joins**
  - No database constraints
  - No aggregation functions (count, avg...)
- You can **only retrieve 1000 records** per query
- In a query, you can **only filter on one column** for inequality
- Transactions only available in entity groups
- You can only update an entity once in a transaction

jeudi 12 novembre 2009

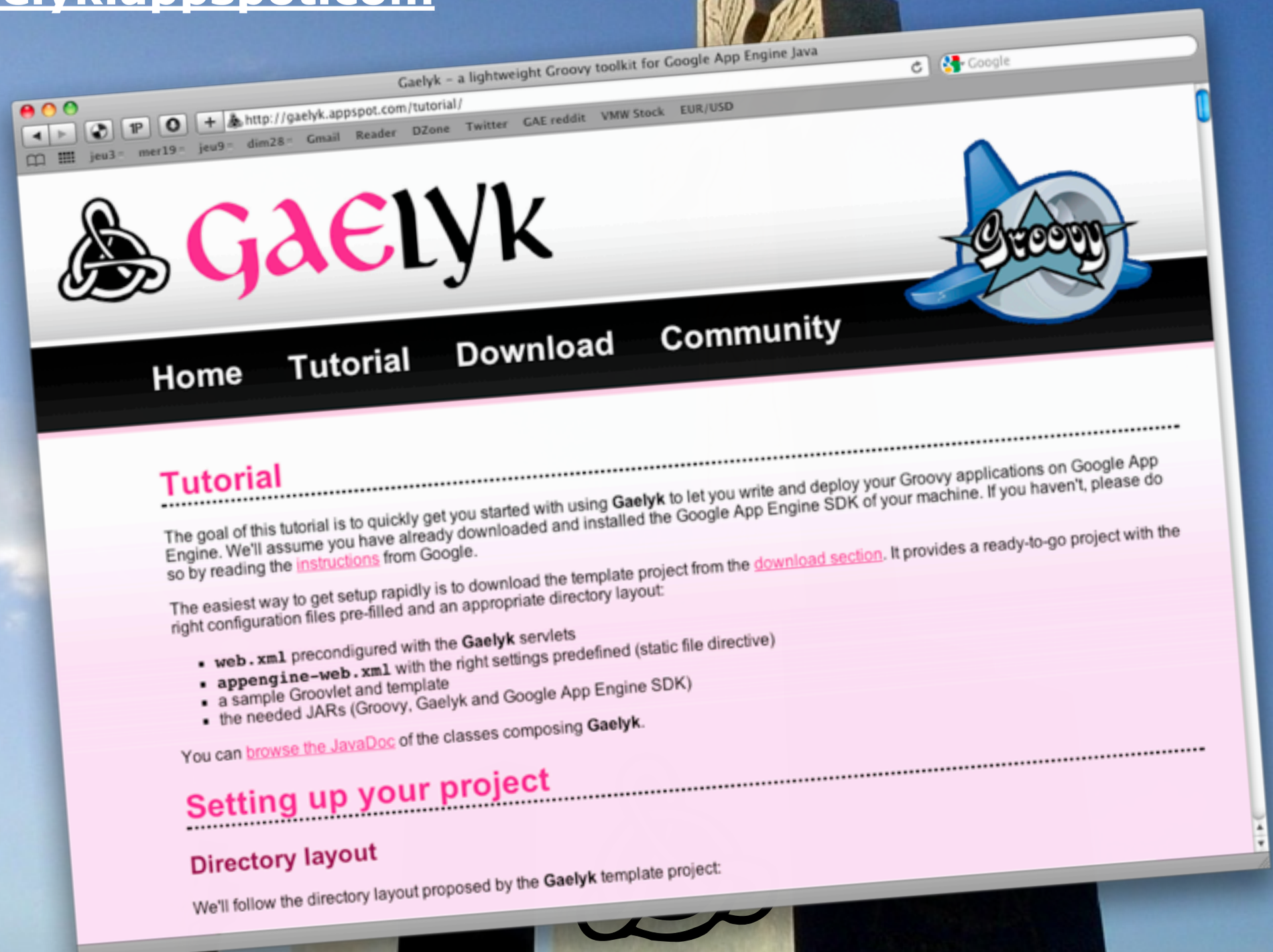
Gaelyk

Gaelyk

http://gaelyk.appspot.com

Gaelyk

http://gaelyk.appspot.com



Why Groovy?

- **Groovy** is a **dynamic language** for the JVM
  - very flexible, malleable, expressive and **concise** syntax
  - easy to learn for Java developers
    - deriving from the Java 5 grammar
  - provides powerful APIs to simplify the life of developers
    - possibility to **dynamically enrich existing APIs**
  - support for **Groovlets**
  - features its own **template engine**

- We worked with the Google App Engine Java team before the official launch of the platform, to ensure Groovy would run well on this new environment

Gaelyk

- Gaelyk is a **lightweight Groovy toolkit** on top of the Google App Engine Java SDK

- Gaelyk builds on Groovy's servlet support
  - **Groovlets**: Groovy scripts instead of raw servlets!
  - **Groovy templates**: JSP-like template engine
  - Both allow for a clean separation of views and logic

- Gaelyk provides several **enhancements** around the GAE Java SDK to make life easier, thanks to Groovy's dynamic nature

First steps...

- Go to http://gaelyk.appspot.com
- Download the template project
- Put your first Groovlet in /WEB-INF/groovy
- And your templates at the root
- And you're ready to go!
- Launch dev_appserver.sh
- Go to http://localhost:8080/

The web.xml

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee" version="2.5">
  <servlet>
    <servlet-name>GroovletServlet</servlet-name>
    <servlet-class>groovyx.gaelyk.GaelykServlet</servlet-class>
  </servlet>
    <servlet>
    <servlet-name>TemplateServlet</servlet-name>
    <servlet-class>groovyx.gaelyk.GaelykTemplateServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>GroovletServlet</servlet-name>
    <url-pattern>*.groovy</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>TemplateServlet</servlet-name>
    <url-pattern>*.gtpl</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
    <welcome-file>index.gtpl</welcome-file>
  </welcome-file-list>
</web-app>
```

Shortcuts

- **Variables available**
  - request / response
  - context / applicaiton
  - session
  - params
  - header
  - out / sout / html
- **Methods available**
  - include / forward
  - print / println
- **Google services**
  - datastoreService
  - memcacheService
  - urlFetchService
  - mailService
  - userService
  - user
  - defaultQueue
  - queues
  - xmppService

jeudi 12 novembre 2009

Sending emails with Gaelyk

```
mailService.send to: 'admin@example.com',
     from: 'user@example.com',
     subject: 'Hello World',
     htmlBody: '<bold>Hello</bold>'
```

...compared to Java

```
Properties props = new Properties();
Session session = Session.getDefaultInstance(props, null);

String msgBody = "...";

try {
    Message msg = new MimeMessage(session);
    msg.setFrom(new InternetAddress("admin@example.com", "Admin"));
    msg.addRecipient(Message.RecipientType.TO,
                     new InternetAddress("user@example.com", "Mr. User"));
    msg.setSubject("Your Example.com account has been activated");
    msg.setText(msgBody);
    Transport.send(msg);
} catch (AddressException e) {}
} catch (MessagingException e) {}
```

A groovlet

- Instead of writing full-blown servlets, just write Groovy scripts (aka Groovlets)

```
def numbers = [1, 2, 3, 4]
def now = new Date()

html.html {
    body {
        numbers.each { number -> p number }
        p now
    }
}
```

A template

```
<html>
    <body>
        <p><%
            def message = "Hello World!"
            print message %>
        </p>
        <p><%= message %></p>
        <p>${message}</p>
        <ul>
        <% 3.times { %>
            <li>${message}</li>
        <% } %>
        </ul>
    </body>
</html>
```

Accessing the datastore

- Direct interaction with the low-level datastore APIs

```
import com.google.appengine.api.datastore.Entity

Entity entity = new Entity("person")

// subscript notation, like when accessing a map
entity['name'] = "Guillaume Laforge"

// normal property access notation
entity.age = 32

entity.save()
entity.delete()

datastoreService.withTransaction {
    // do stuff with your entities
    // within the transaction
}
```

Querying to be improved...

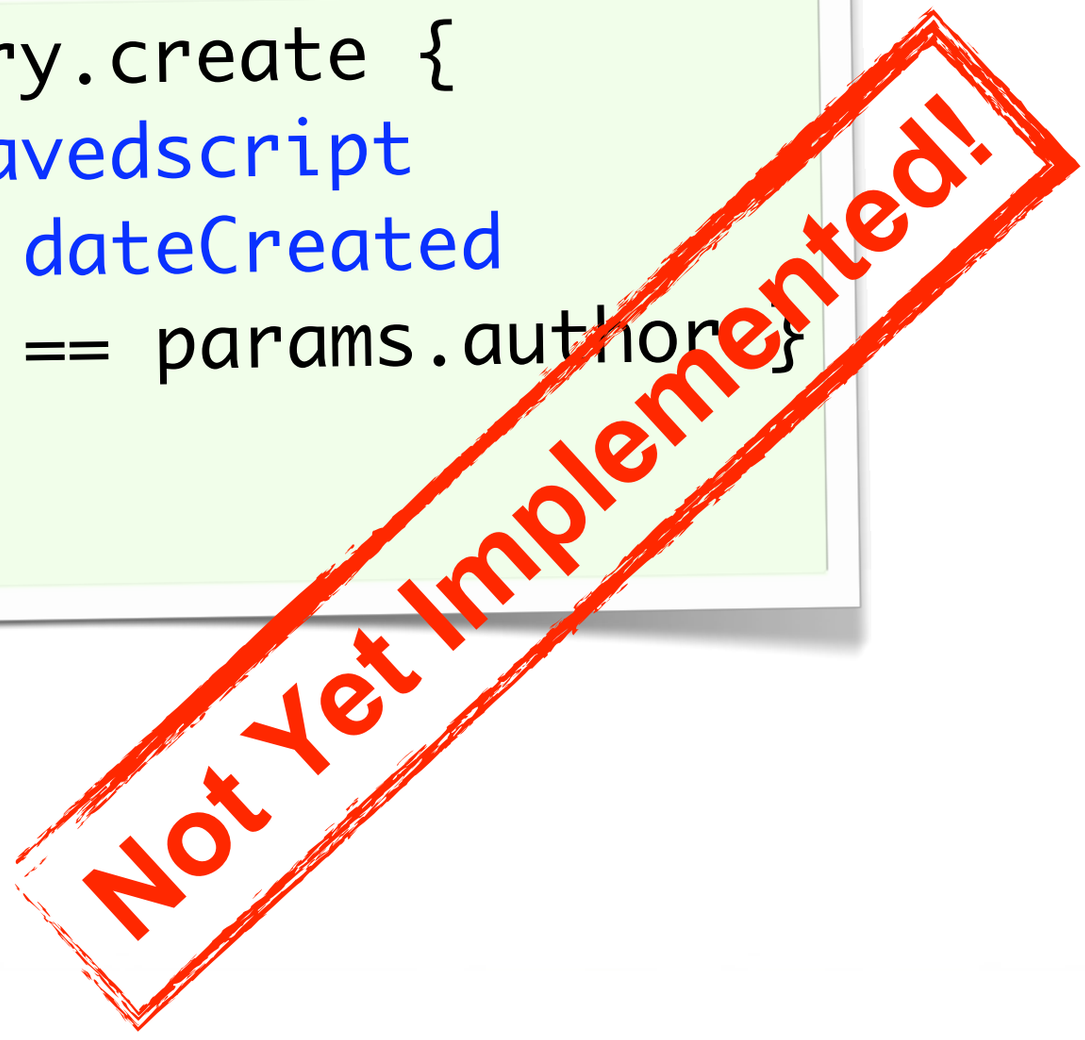
```
import com.google.appengine.api.datastore.*
import static com.google.appengine.api.datastore.FetchOptions.Builder.*

// query the scripts stored in the datastore
def query = new Query("savedscript")

// sort results by descending order of the creation date
query.addSort("dateCreated", Query.SortDirection.DESCENDING)

// filters the entities so as to return only scripts by a certain author
query.addFilter("author", Query.FilterOperator.EQUAL, params.author)

PreparedQuery preparedQuery = datastoreService.prepare(query)

// return only the first 10 results
def entities = preparedQuery.asList( withLimit(10) )
```

...into something groovier?

```
def entities = Query.create {
    select from: savedscript
    sort DESC, on: dateCreated
    where { author == params.author }
    limit 10
} as List
```

...into something groovier?

```
def entities = Query.create {
    select from: savedscript
    sort DESC, on: dateCreated
    where { author == params.author }
    limit 10
} as List
```



Task queue API

```
// access a configured queue using the subscript notation
queues['dailyEmailQueue']

// or using the property access notation
queues.dailyEmailQueue

// you can also access the default queue with:
queues.default
defaultQueue

// add a task to the queue
queue << [
    countdownMillis: 1000, url: "/task/dailyEmail",
    taskName: "Send daily email newsletter",
    method: 'PUT', params: [date: '20090914'],
    payload: content
]
```

Jabber / XMPP support (1/3)

- Sending instant messages

```
String recipient = "someone@gmail.com"

// check if the user is online
if (xmppService.getPresence(recipient).isAvailable()) {
    // send the message
    def status = xmppService.send(to: recipient,
                                  body: "Hello, how are you?")

    // checks the message was successfully
    // delivered to all the recipients
    assert status.isSuccessful()
}
```

Jabber / XMPP support (2/3)

- Sending instant messages with an XML payload

```
String recipient = "service@gmail.com"

// check if the service is online
if (xmppService.getPresence(recipient).isAvailable()) {
    // send the message
    def status = xmppService.send to: recipient, xml: {
        customers {
            customer(id: 1) {
                name 'Google'
            }
        }
    }

    // checks the message was successfully delivered to the service
    assert status.isSuccessful()
}
```

Jabber / XMPP support (2/3)

- Sending instant messages with an XML payload

```
String recipient = "service@gmail.com"

// check if the service is online
if (xmppService.getPresence(recipient).isAvailable()) {
    // send the message
    def status = xmppService.send to: recipient, xml: {
        customers {
            customer(id: 1) {
                name 'Google'
            }
        }
    }

    // checks the message was successfully delivered to the service
    assert status.isSuccessful()
}
```

```
<customers>
    <customer id='1'>
        <name>Google</name>
    </customer>
</customers>
```

Jabber / XMPP support (3/3)

- Receving incoming instant messages
  - Configure the XmppServlet in web.xml
  - Add the inbound message service in appengine-web.xml

```
// get the body of the message
message.body

// get the sender Jabber ID
message.from

// get the list of recipients Jabber IDs
message.recipients

// if the message is an XML document instead of a raw string message
if (message.isXml()) {
    // get the raw XML
    message.stanza

    // get a document parsed with XmlSlurper
    message.xml
}
```

What's coming next?

- Add more sugar around...
  - The Memcache service
  - The incoming email support in GAE SDK 1.2.6
  - The Datastore query system
    - SQL-like DSL
    - dynamic finders
- More generally...
  - Anything that'll come up in upcoming GAE SDK versions

Summary

- Easy access to a cloud solution
  - Deploying Java apps, as easily as you would with PHP
- Familiar to Java folks
  - Your good old Servlet centric webapps style

- Pretty cheap
  - You need a high-trafficed website to reach the quotas
- Gaelyk provides a simplified approach to creating Servlet centric webapps in a productive manner
  - Leveraging Groovy's servlet / template support and dynamic capabilities

Q&A

References:

- http://code.google.com/appengine/
- http://gaelyk.appspot.com/
- http://groovy.codehaus.org/
- http://grails.org/

Images utilisées dans cette présentation

- Nuages
  - http://www.morguefile.com/archive/display/627059
  - http://www.morguefile.com/archive/display/625552
  - http://www.morguefile.com/archive/display/629785
- Le village dans les nuages
  - http://www.collectoy.com/photoBDD/Disque/Le%20village%20dans%20les%20nuages%20(%20face%20).jpg
- Duke ok GAE
  - http://code.google.com/images/duke-on-gae.jpg
  - http://weblogs.java.net/blog/felipegaucho/archive/ae_gwt_java.png
- Python logo : http://python.org/images/python-logo.gif
- Gaelyc cross with clouds : http://www.morguefile.com/archive/display/37889
- Speed limit : http://www.morguefile.com/archive/display/18492
- Warehouse : http://www.morguefile.com/archive/display/85628
- Snow foot steps : http://www.flickr.com/photos/robinvanmourik/2875929243/
- Sugar : http://www.flickr.com/photos/ayelie/441101223/sizes/l/